Please replace the paragraph at page 30, lines 13-19 with the following.

--Figure 2 is a diagrammatic illustration of the network and platform components of the nMCI Interacts system, including: the Customer workstation 20; the Demilitarized Zone 17 (DMZ); Web Servers cluster 24; the MCI Intranet Dispatcher/Proxy Servers/application server cluster 30; and the MCI [Intranet Application] servers 40, warehouses, legacy systems, etc.--

Please replace the paragraph at pages 32, line 27 to page 33, line 28 with the following.

--As will be hereinafter described in greater detail, a networkMCI Interact session is designated by a logon, successful authentication, followed by use of server resources, and logoff. However, the world-wide web communications protocol uses HTTP, a stateless protocol, each HTTP request and reply is a separate TCP/IP connection, completely independent of all previous or future connections between the same server and client. The present invention is implemented with a secure version of HTTP such as S-HTTP or HTTPS, and preferably utilizes the SSL implementation of HTTPS. The preferred embodiment uses SSL which provides a cipher spec message which provides server authentication during a session. The preferred embodiment further associates a given HTTPS request with a logical session which is initiated and tracked by a "cookie jar server" 28 to generate a "cookie" which is a unique server-generated key that is sent to the client along with each reply to a HTTPS request. The client holds the cookie and returns it to the server as part of each subsequent HTTPS request. As desired, either the Web servers 24, the cookie jar server 28 or the Dispatch Server 26, may maintain the "cookie jar" to map these keys to the associated session. A separate cookie jar server 28, as illustrated in Figure 2 has been found desirable to minimize the load on the dispatch server 26. A new cookie will be generated when the response to the HTTPS request is sent to the client. This form of session management also

functions as an authentication of each HTTPS request, adding an additional level of security to the overall process.--

Please replace the paragraph at page 41, lines 10-23 with the following.

--COAppFrame 56a, 56b is a desktop window created and used by a COApp to contain its user interface. The COAppFrame 56a, 56b is a separate window from the Web browser 14. Generally, the COAppFrame 56a, 56b has a menu, toolbar, and status bar. The COAppFrame's attachToViewArea ( ) method may be used to paste a COView object 58a, 58b, 58c into a COAppFrame 56a, 56b. The COView class is an extension of java.awt.Panel. It provides a general purpose display space and container for an application's visual representation. Application classes typically extend the COView class to implement their presentation logic. COApp may use none, one, or many COAppFrames 56a, 56b.--

Please replace the paragraph at page 150, lines 16 to 28 with the following.

--Figure 25(b) illustrates a diagram depicting the execution of a transaction by the SI application server 36 with each bubble representing a separate thread. The following itemized scenario describes the sequence of events in detail with each number in the scenario associated with the numbers in Figure 25(b). First, at step 2501, the SI Application Server instantiates and starts the Transaction Manager 2320 in a separate thread. The SI Application Server then instantiates and starts the Transaction Server 2500 in a separate thread at step 2502. The SI Application Server 36 instantiates and starts the Registry Server in a separate thread at step 2503.--

Please replace the paragraph at page 151, lines 1 to 15 with the following.